

# A Cognitive Simulation Model for Novice Text Entry on Cell Phone Keypads

**Arindam Das**

Computer Science  
York University  
Toronto, Ontario, Canada  
arindam@cse.yorku.ca

**Wolfgang Stuerzlinger**

Computer Science  
York University  
Toronto, Ontario, Canada  
wolfgang@cse.yorku.ca

## ABSTRACT

**Motivation** – To create a cognitive simulation model that predicts text entry performance and learning on cell phone keypads by novice users.

**Research approach** – A programmable cognitive architecture, ACT-R, is used to execute the simulation model. Part of the simulation result is compared with the result of a previous user study.

**Findings/Design** – The proposed model is an a priori model (not tuned to any real user data) that predicts the amount of time spent in finding a key on the keypad and pressing it repeatedly. The predicted amount of time in finding a key differs by 6% and the time between two repeated key-presses of the same key by 27% compared to the results of a previous user study. The model also captures the learning of keypad layout by novice users. Memorization of keypad layout is simulated using task repetition.

**Research limitations/Implications** – This research has several limitations described towards the end of this paper. An important one among them is that the work does not model the impact of visual distracters in the field of view (frontal surface of the handset) on user performance.

**Originality/Value** – This is the first cognitive simulation model of novice user's text entry performance and learning on cell phone keypads.

**Take away message** – This work introduces an a priori cognitive model of text entry by novice users. This forms a basis for systematic exploration of keypad designs for cell phones in shorter time and lower cost.

## Keywords

Novice users, text entry, cognitive architecture, ACT-R.

## INTRODUCTION

Traditionally, in the domain of Human-Computer Interaction, research in evaluating text-entry performance on cell phone keypads has focused on user studies. Most user studies were conducted with expert users as reported in a recent survey (MacKenzie & Soukoreff, 2002). Even though the population of expert cell phone users is large and is currently growing at a high rate, a sizeable segment of potential users are

unskilled in using such handheld devices. Examples include elderly people and some adults (especially in geographical regions where cellular technology hasn't proliferated yet). Unlike expert users these persons are not able to enter text quickly due to lack of practice or other limitations. Instead, they often resort to "hunt and peck" level of text entry and concentrate on completing their task accurately albeit slowly. To date, very little work has been done to address the behaviour of such novice users, one exception being (Pavlovych & Stuerzlinger, 2004).

The user studies reported in (MacKenzie et al, 2002; Pavlovych et al, 2004) have four characteristics in common: first, most of them required paid human subjects; second, they required, in most cases, development of sophisticated software and hardware for the experiments; third, several of them needed substantial time to complete; fourth, many of them were constrained by the keypad layout provided by the manufacturer of the handset used in the experiments. Overall, the studies were labour- and time-intensive. In contrast, to facilitate quick exploration of a great number of new keypad designs at lower cost, this work develops a cognitive simulation model of novice users of text entry that can directly predict user performance.

Our model is based on the cognitive architecture ACT-R (Anderson, Bothell, Byrne, Douglass, Lebiere & Quin, 2004). To our knowledge, ACT-R has only been applied twice in the domain of modelling handheld device interactions: one work involved cell phone menu interaction (St. Amant, Horton & Ritter, 2007) and the other involved the merger of the Keystroke-Level Modelling (KLM) and ACT-Simple modelling techniques (John, Prevas, Salvucci & Koedinger, 2004). Both of them targeted expert users and did not model novice behaviour.

One of our goals of this paper is to model the tasks of the first user study in (Pavlovych et al, 2004), which focused on part of the behaviour of novices in text entry on cell phones. The user study consisted of two parts: One part measured the non-Fitts amount of time that precedes a key press for the first letter of a letter-group containing multiple instances of the same letter in cases when no visual verification of outcome is necessary. By non-Fitts time, we mean the fraction of the user's total

task performance time that remains after subtracting the movement time predicted by Fitts law (Fitts, 1954). The other part measured the non-Fitts amount of time between two repeated key-presses of the same key.

A secondary aim of this work is to capture the learning process of novice users using a simple memory model. In order to fulfil that aim, our simulation model predicts the non-Fitts amount of time that precedes a key press for the first letter of two letter-groups containing the same first letter.

### ACT-R COGNITIVE ARCHITECTURE

The ACT-R cognitive architecture (Anderson et al) is a computer-based simulation framework that embodies the theories of cognition, perception and motor behaviour. It enables the creation of models that simulate the temporal (and often simultaneous) progression of cognitive processes, such as attentional changes and memory retrieval, as well as motor processes, such as movement of fingers. It has been the core for several models in Human-Computer-Interaction (Ritter & Young, 2001).

ACT-R can be broadly divided into three modules: perceptual-motor, memory and goal. The perceptual-motor modules are responsible for interfacing with the real world (more specifically, with a simulation of the real world). The perceptual-motor modules in ACT-R are the motor, vision, audition and speech. The memory modules in ACT-R are declarative and procedural memory. The declarative memory module stores facts represented by instances (termed chunks) of abstract data types (termed chunk-types) with attributes (termed slots) in them. Slots may contain values. The chunk-types are both built-in and modeller-defined. The procedural memory module stores the production rules. The production rules represent the knowledge about how a human does things (i.e. procedural knowledge): for example, the knowledge about how to type the letter "c" on a handheld phone keypad. The goal module keeps track of current goals and intentions. All these modules are assumed to be associated with distinct cortical regions. For details on the mappings of the modules to cortical regions, the reader is referred to (Anderson et al).

Usually, an ACT-R model, representing a human user, consists of chunks of modeller-defined chunk-types and production rules. In order to simulate the interaction between the human user and a device, the production rules are executed. During execution, these rules interact with different modules through their respective mediators (termed buffers) and utilize the information in the chunks (placed in the buffers by the modules) to accomplish the desired goal.

### BEHAVIOR OF NOVICE USERS

We informally interacted with few right-handed elderly people in a shopping mall. None of them had performed

text messaging on any kind of handheld device before. Some of them did not possess any kind of handheld communication device. We handed a (non-working) Nokia 5190 handset to each of them and asked them to key in the sequence of letters "ccc zz i yy kkk ss f rrr", i.e. "press the button with 'c' three times, the button with 'z' two times" and so on. Each user keyed-in the sequence only once. (A similar handset and methodology was used in the user studies presented in (Pavlovych et al, 2004)).

After having visually observed the finger movement of the users and having discussed with them the way they used the handset, we extracted the following observations about novice user behaviour:

1. When a novice user holds the handset for the first time for texting, her right thumb would roughly be at the location (3, 0) shown as the grey "start pos" oval on the right side of the ScrollUp key in Figure 1.
2. After pressing a key from the 2nd, 3rd or 4th row of Figure 1, a user seems to recoil the thumb to some location in column 3 of Figure 1.
3. A user might be slowed down in her visual search for the target letter due to the presence of several visual distracters (numbers, letters and other symbols) scattered over the frontal surface of the handset.
4. Users seem to be keeping count of the number of presses made while performing repeated presses of the same key. This is the same behaviour that was also observed in a user study in previous work (Pavlovych et al, 2004).

### THE COGNITIVE SIMULATION MODEL

Our cognitive simulation model utilizes four modules of ACT-R 6.0 - motor, vision, memory, and goal modules. (i) We use the motor module to model the interaction of the right-hand thumb with the keys on the keypad. Figure 1 shows the model of the keypad that our cognitive model interacts with through the motor module. (ii) We use the vision module to model the visual attention of the stimuli (letters). Figure 2 represents the visual scene in the external environment that our cognitive model interacts with through the vision module. It coarsely mimics the frontal surface of the Nokia 5190 handset built with the abilities of ACT-R 6.0 field-of-view builder. (iii) The memory model utilizes the declarative memory module for memorization. (iv) We use the goal module to keep track of the current state of the execution.

### Adapting the ACT-R Framework

The motor module of the default ACT-R framework contains the model of a QWERTY computer keyboard (more specifically an Apple Extended Keyboard II) as the only model of an input device. In order to support the development of our cognitive model for text entry on cell phones, we added a model for the keypad interface of a Nokia 5190 handset (as used in (Pavlovych et al, 2004)) to the motor module of ACT-R

and also added certain motor movement styles as follows:

(i) We created a virtual grid of key locations and the start and the recoil positions for the right thumb. Figure 1 shows this grid, with four columns and six rows. Columns 0 to 2 contain the locations of the keys, whereas column 3 contains the start and the recoil positions of the right-hand thumb. Although the recoil position might vary and hence affect the movement time predicted by Fitts' law, our assumption of a fixed recoil position is still valid for this work since we are interested only in the non-Fitts component of user's total task performance time. We further assumed that a) all the keys on the keypad are of the same size, b) the width of a key is one "key unit", c) the horizontal and vertical distance between adjacent keys on the keypad is one "key unit", and d) that the user is right-handed (holds the handset in her right hand) and uses only the thumb to press keys.

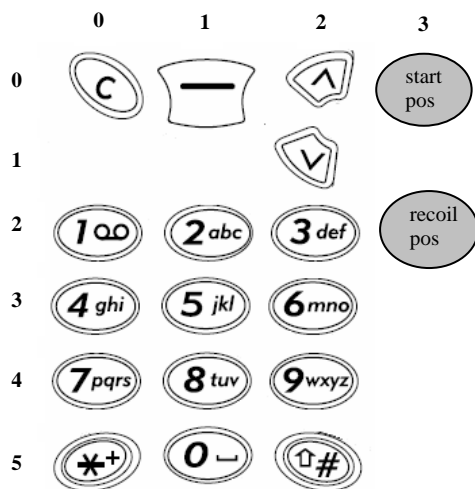


Fig. 1. Virtual grid for the Nokia 5190 keypad.

(ii) We created a new movement style called thumb-recoil-to-location that models the movement of the right-hand thumb from a key to the recoil location (3, 2). The grammar for the new style is as follows:

```
+manual>
ISA      thumb-recoil-to-location
hand     right
finger   thumb
to-loc   location in virtual grid
```

(iii) The default *Peck* movement style of ACT-R (a directed movement of a finger to a new location followed by a keystroke, all as one continuous movement) may be considered sufficient for text entry modelling. However, this style was developed for

computer keyboards where only one letter is mapped to one key. Since a single key on a typical phone keypad maps to multiple characters we were forced to extend the ACT-R system in order to allow the modeller to specify the location of the target key as well as the character the cognitive model would be pecking for. We named the new movement style *peck-to-location-for-char*. The grammar for the extended style is as follows:

```
+manual>
ISA      peck-to-location-for-char
hand     right
finger   thumb
to-loc   location in virtual grid
for-char string
```

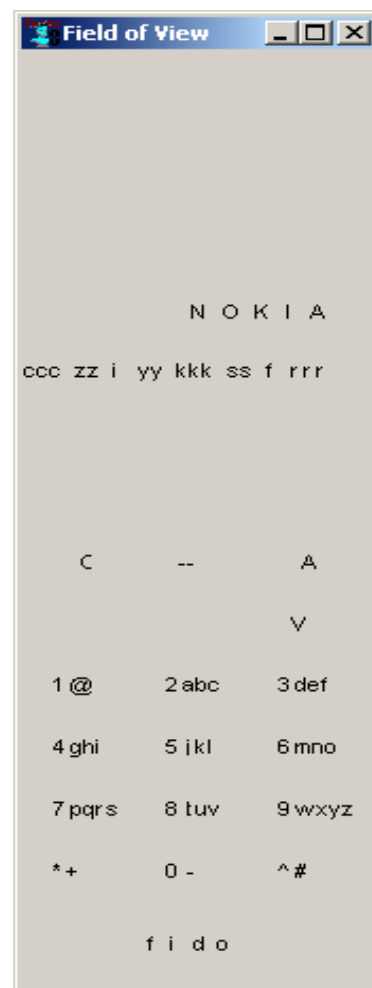


Fig. 2. Visual scene (field of view) for the model.

(iv) The default *Punch* movement style of ACT-R (a down-stroke directly followed by an upstroke of a finger, for pressing a key that is already directly below

the finger) was originally developed for the home keys (recoil / resting positions of the fingers) on a computer keyboard. In our case, however, punch can be executed on any key and even more importantly one or more “punches” of a key are *necessary* to enter secondary, tertiary, etc. letters of each key. We therefore extended the default movement style to allow the modeller to specify the character to be punched as well. We named the new style `punch-for-char`. The grammar for the extended style is as follows:

```
+manual>
  ISA      punch-for-char
  hand     right
  finger   thumb
  for-char string
```

We also updated the ACT-R motor framework to remove the Fitts law time from the task performance time during simulation. We did this since we are only interested in the non-Fitts time.

### Implementation

Our model uses a single modeller-defined chunk-type. The chunks created from this chunk-type helps the model to keep track of the state of search of a letter on the displayed phrase as well as on the keypad, the last letter searched and found, the current letter being searched, the location of the current letter on the keypad and the location of the current letter on the displayed phrase.

We represent the procedural knowledge of our cognitive model using the following nineteen production rules:

- *seek-location-of-first-char-on-phrase* finds the position of the first letter of the displayed phrase in the field of view.
- *switch-attention-on-phrase* shifts the visual attention to the position of the current letter that has been found on the displayed phrase.
- *extract-stimulus-on-phrase* encodes the attended letter on the displayed phrase so that the letter becomes accessible to the model.
- *char-does-not-equal-last-char-on-phrase* matches if the current letter found is not same as the last letter. If a match occurs, it attempts to retrieve the keypad position of the current letter from the declarative memory. If the retrieval is successful, the retrieval buffer (associated with declarative memory) is filled up with the chunk containing the letter and its coordinates on keypad. If the retrieval is not successful, the retrieval buffer becomes empty. This rule is tried only for the retrieval of the keypad position of the first letter of each letter-group.
- *recall-location-on-keypad* matches if the keypad coordinates of the current letter (that has just been encoded from the displayed phrase) is same as the

information present in the retrieval buffer and fails to match if it doesn't. If the match occurs, the model will execute a motor action directly, without any attention shift, to enter the letter.

- *cannot-recall-location-on-keypad* matches if the keypad coordinates of the current letter (that has just been encoded from the displayed phrase) is not same as the information present in the retrieval buffer (more specifically when the retrieval buffer is empty). If the match occurs, it will lead to the shift of visual attention, to the keypad area, for the current letter.
- *char-equals-last-char-on-phrase* matches when the current letter found is same as the last letter. If the match occurs, a motor action is carried out.
- *seek-location-of-char-on-keypad* finds the position of the current letter (that has just been encoded from the displayed phrase) on the keypad when the letter's position on the keypad cannot be recalled from declarative memory.
- *switch-attention-on-keypad* shifts the visual attention to the position of the current letter that has been found on the keypad.
- *extract-stimulus-on-keypad* encodes the attended letter on keypad so that the letter becomes accessible to the model.
- *do-peck-for-first-char-of-the-phrase* executes a peck movement for the first letter of the phrase.
- *prepare-for-thumb-recoil* gets the model ready for recoiling the right thumb.
- *do-thumb-recoil-before-peck-for-next-char* enables the model to recoil its right thumb to the recoil home location (3, 2) shown in Figure 1.
- *do-peck-for-next-char-of-the-phrase-after-thumb-recoil* enables the model to execute a peck movement for the relevant letter of the phrase (except the first letter of the phrase). In our case, this rule will apply to the first letter of every letter-group (except for the first group).
- *do-punch-when-char-equals-last-char* enables the model to execute a punch movement when the letter to be entered is same as the last letter entered.
- *get-location-of-current-char-on-phrase* retrieves the location of the current letter in the displayed phrase from declarative memory. Note that this production only helps in getting the thread of control back to the phrase from the keypad after each letter is entered. This production/transfer between foci of attention cannot be avoided and adds an overhead of 50 ms for every letter entered.
- *seek-location-of-next-char-on-phrase* attempts to find the position of the next letter of the phrase.
- *end-of-phrase-not-reached-yet* is fired when the position of a new letter in the phrase has been found.

In that case, the visual attention is shifted to the position of the newly found letter. At this point, the execution continues.

- *end-of-phrase-reached* is fired when there are no more letters left to be read in the phrase. At this point, the execution stops.

Similar to (Pavlovych et al, 2004), our model is constructed in such a way that it avoids repeated key presses required to scroll for a letter. For example, in the character sequence “ccc zz i yy kkk ss f rrr”, when a user presses the key with letter ‘z’ for the first time (the fourth key-press), ‘z’ gets displayed right away and when the same key is pressed seventh time, the letter ‘y’ appears in the display. We do this in order to stay compatible with the user study in (Pavlovych et al, 2004) that we model; however we point out that our model can easily simulate the effect of scrolling.

### COGNITIVE SIMULATION EXPERIMENTS

We executed the model in virtual time in two different sessions on ACT-R 6.0 (with our aforementioned modifications included) leaving the ACT-R model parameters and constants at their default values. The model was not tuned specifically to any user data. We describe the behaviour of the model in two sessions next.

#### First Execution Session

In the first session, the model simulated the task of finding a key on the keypad and pressing it repeatedly (similar to first user study in (Pavlovych et al, 2004)). The model of the novice user behaved as follows:

1) The phrase to be entered by the user is “ccc zz i yy kkk ss f rrr” and displayed near the top of the phone as illustrated in figure 2. Let us assume that, in between the displayed phrase and keypad in figure 2, there is a screen area where a character appears when keyed in by the user using the keypad. The user copies only the letters of the displayed phrase to the screen area. She does not copy spaces, as the spaces serve only as visual separation of the letter groups. 2) Initially, the user holds the handset in her right hand with her right thumb on location (3, 0), the start position of figure 1. 3) First, the user searches the displayed phrase and finds the first letter of a letter group. Second, she tries to recall the position of the letter on the keypad and she fails. Third, she searches visually and finds the letter on the keypad. Fourth, she *pecks* the key containing the letter with her right thumb. Fifth, she looks back to the displayed phrase to pick up the next letter. Sixth, if the next letter is the same as the last letter, she *punches* the same key where her thumb was on last time. Finally, once she finishes entering all the letters in a letter group, she recoils her thumb to the location (3, 2) shown in figure 1. She continues repeating the steps for all the letter groups until finished.

The following results were obtained: a) The non-Fitts time preceding the key-press for the first letter of the

first letter-group ‘ccc’ was 970 ms. b) The non-Fitts time preceding the key-press for the first letter of the remaining letter groups was 1420 ms. c) The non-Fitts time between the key-presses for the first and the second letter in a letter-group was 400 ms. d) The non-Fitts time between the key-presses to enter consecutive letters other than the first letter in a letter-group was 235 ms.

The increase in the non-Fitts time from (a) to (b) above is mainly due to the recoiling movement of the thumb just before the first letter of a new letter-group (other than the first group) is pressed.

The comparison with real-world data from user study (Pavlovych et al, 2004) is summarized in Table 1. The mean time from ACT-R simulation is about 6% larger than the time preceding the key press of the first letter and about 27% larger than the time between two repeated key presses. The user study in (Pavlovych et al, 2004) conjectures that the user keeps count of the number of presses made while performing repeated presses of the same key and the time to keep such count is the non-Fitts time between two repeated key-presses. Comparing the mean time (of the user study [272 ms] versus the simulation [345 ms]) between two repeated key presses in Table 1, we deduce that our cognitive model is basically in agreement with the time spent in keeping such a mental count.

Non-Fitts time component	User Study (Pavlovych et al, 2004) (ms)	ACT-R simulation (ms)
Mean Time preceding key press of first letter	1285	1363.75
Mean Time between two repeated key presses	272	345

Table 1. Mean non-Fitts time across key press actions.

There are two potential explanations for the differences between the real-world data and our prediction shown in Table 1: (a) the results from the user study in (Pavlovych et al, 2004) were averages of around 900 task repetitions in total by 12 participants (i.e. around 75 task repetitions per participant). These repetitions will have induced some learning in the participants, which in turn will lower average performance times somewhat. (b) 25% of the participants of the user study were frequent users of text messaging. The (relatively better) performance of those users may also have lowered the performance averages.

All that said, we emphasize that we can use this model as the basis for comparing different keyboard layouts since relative performance is often sufficient to guide design.

#### Second Execution Session

In the second session, the model simulated learning of the keypad layout by a novice user. The model behaved

as described below. The setup stayed exactly the same as in first session, only the phrase to be entered by the model changed:

1) The phrase to be entered by the user in this session is “ccc ee u ee rrr hh u rrr” and it is displayed near the top of the phone. 2) First, the user searches the displayed phrase and finds the first letter of a letter group. Second, she tries to recall the position of the letter on the keypad. If she sees the letter in the phrase for the first time, the recall fails. Otherwise, she recalls successfully. Third, if the recall fails, she searches the keypad to find the letter on it and then *pecks* the key containing the letter with her right thumb. If the recall succeeds, she does not spend time searching the keypad for the letter; rather, she directly *pecks* the key (containing that letter) since she already remembers its position on the keypad. Fourth, she looks back to the displayed phrase to pick up the next letter. Fifth, if the next letter is the same as the last letter, she *punches* the same key where her thumb was on last time. Finally, once she finishes entering all the letters in a letter group, she recoils her thumb to the location (3, 2) shown in figure 1. She continues repeating the steps for all the letter groups until finished.

The comparison of the cognitive time for entering the first letter of each group (except the first group ccc) is summarized in Table 2. Since keying-in of an already memorized first letter of a group would involve a prior thumb recoil (in case of the phrase we used), we left the first group out of our comparison as it is not preceded by a thumb recoil. This was done to keep the comparison simple. While the previous user study (Pavlovych et al, 2004) reported an average cognitive time of 1285 ms for keying-in the first letter of a group (note that small amount of learning got induced because of few non-novice participants as explained towards the end of the last subsection), our ACT-R model took 1420 ms to enter the first letter of the first ‘ee’ group (i.e. the 4th character) and 1185 ms to enter the first letter of second ‘ee’ group (i.e. the 7th character). Similar decrease in text entry time occurred for the first letter of every second (repeated) group of ‘u’ and ‘rrr’ owing to learning by the model. From Table 2, it is apparent that the simulation result nicely brackets data from the previous user study in (Pavlovych et al, 2004).

Novice user study (short learning) (Pavlovych et al, 2004) (ms)	ACT-R model (with thumb recoil, before learning) (ms)	ACT-R model (with thumb recoil, after learning) (ms)
1285	1420	1185

Table 2: Mean cognitive time for press of the first letter of each letter group.

## LIMITATIONS OF THE COGNITIVE MODEL

There are several limitations of our model: i) It does not capture learning as a natural gradual process that would follow the power law of learning curve; this work just demonstrates that the model is capable of learning through instantaneous memorization. ii) It does not capture any potential forgetting effects that could occur due to lack of rehearsals over time. iii) It does not take into account the noise from several visual distracters (numbers, letters and other symbols) on the frontal surface of the handset that might slow down the visual search for the target letter. iv) It does not model the eye movement time of the user while shifting attention from the letter in the displayed phrase to that on the keypad. This is because the vision module in ACT-R is an attentional system; it does not take into account the time spent in eye movement. v) It does not mimic the borders of the buttons of a keypad in its visual scene as found in figure 2. vi) It does not model potential errors in text entry.

## FUTURE WORK

We plan to address some of the aforementioned limitations in our future work. For example, we can improve our model to account for the visual distracters present in the field of view. In this regard, (Byrne, 2001) could act as the closest cue for the research. We also plan to undertake a user study with purely novice users for validating the predictions about learning behaviour. Besides, one research direction could be to explore the possibility of creating a cognitive simulation model that would help predict the improvement in spatial learning due to effort-inducing mobile keypad interfaces and then compare it with the real-world data from a recently concluded user study (Cockburn, Kristensson, Alexander & Zhai, 2007). The work (Ehret, 2002) could act as the closest cue for this investigation. Another research avenue could involve leveraging the lessons learnt in (Grossman, Dragicevic & Balakrishnan, 2007) on accelerating learning strategies and explore them in the domain of handheld keypad interfaces, in general, through cognitive simulation models.

## CONCLUSION

In this article we described a new a priori cognitive simulation model of novice users that predicts the non-Fitts time spent to find a key on a cell phone keypad and pressing it repeatedly. The model is also able to predict learning effects for this. We used the cognitive architecture ACT-R to execute our model. In the process of developing the model, we added new motor movement styles to the architecture, thereby enhancing the modelling support of its motor module. We believe that ACT-R models will favourably supplement our future user studies of novice text entry on handheld devices by providing quick a priori performance estimates of keypad design at significantly lower cost,

also facilitating systematic exploration of the design space. Besides, new such models can also help us to better explain data obtained in previous user studies on novice users.

#### ACKNOWLEDGMENTS

We would like to thank Scott MacKenzie of York University, Dan Bothell of CMU and Frank Ritter of Penn State University.

#### REFERENCES

- Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., and Quin, Y. (2004). An integrated theory of the mind. *Psychological Review*, 111(4), 1036–1060.
- Byrne, M. D. (2001). ACT-R/PM and menu selection: Applying a cognitive architecture to HCI. *International Journal of Human-Computer Studies*, 55, 41-84.
- Cockburn, A., Kristensson, P. O., Alexander, J., and Zhai, S. (2007). Hard Lessons: Effort-Inducing Interfaces Benefit Spatial Learning, in *Proceedings of CHI '07* (New York), ACM Press, 1571-1580.
- Ehret, B. (2002). Learning Where to Look: Location Learning in Graphical User Interfaces, in *Proceedings of CHI '02* (New York), ACM Press, 211-218.
- Fitts, P. M. (1954). The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, 47(6), 381-391.
- Grossman, T., Dragicevic, P., and Balakrishnan, R. (2007). Strategies for Accelerating On-line Learning of Hotkeys, in *Proceedings of CHI '07* (New York), ACM Press, 1591-1600.
- John, B. E., Prevas, K., Salvucci, D. D., and Koedinger, K. (2004). Predictive human performance modeling made easy, in *Proceedings of CHI '04* (New York), ACM Press, 455–462.
- MacKenzie, I. S., and Soukoreff, R. W. (2002). Text Entry for Mobile Computing: Models and Methods, Theory and Practice. *Human-Computer Interaction*, 17, 147-198.
- Pavlovych, A., and Stuerzlinger, W. (2004). Model for non-Expert Text Entry Speed on 12-Button Phone Keypads, in *Proceedings of CHI '04* (New York), ACM Press, 351-358.
- Ritter, F. E., and Young, R. M. (2001). Embodied models as simulated users: Introduction to this special issue on using cognitive models to improve interface design. *International Journal of Human-Computer Studies* 55, 1–14.
- St. Amant, R., Horton, T. E., and Ritter, F. E. (2007). Model-Based Evaluation of Expert Cell Phone Menu Interaction. *ACM Transactions on Computer-Human Interaction*, Vol. 14, No. 1, 1–24.